Systèmes d'inférence floue auto-évolutifs : apprentissage incrémental pour la reconnaissance de gestes manuscrits

Abdullah Almaksour, Eric Anquetil

Systèmes d'inférence floue auto-évolutifs : apprentissage incrémental pour la reconnaissance de gestes manuscrits

Abdullah Almaksour, Eric Anquetil

INSA de Rennes, Avenue des Buttes de Coesmes, F-35043 Rennes
CNRS, UMR IRISA, Campus de Beaulieu, F-35042 Rennes
Université Européenne de Bretagne, France
{Abdullah.Almaksour, Eric.Anquetil}@irisa.fr

RÉSUMÉ. *Nous présentons dans ce papier une nouvelle méthode pour la conception de moteurs de reconnaissance de gestes manuscrits personnalisables et auto-évolutifs, c'est-à-dire capables de s'adapter au style d'écriture et aux habitudes de chacun, sans toutefois nécessiter de période d'apprentissage fastidieuse. Nous utilisons une approche d'apprentissage incrémental de classifieurs basés sur les systèmes d'inférence floue de type Takagi-Sugeno. Cette approche comprend d'une part, une adaptation des paramètres linéaires associés aux conclusions des règles en utilisant la méthode des moindres carrés récursifs, et d'autre part, un apprentissage incrémental des prémisses de ces règles afin de modifier les fonctions d'appartenance suivant l'évolution de la densité des données dans l'espace de classification.*

ABSTRACT. *We present in this paper a new method for the design of customizable self-evolving handwriting recognition systems, which are able to adapt to writing style and needs of each writer, without require time-consuming re-learning process. The presented approach is based on first-order Takagi-Sugeno fuzzy inference system. This approach involves first an incremental clustering and adaptation of the premise part of the system, and secondly, an incremental learning of the linear consequents parameters of the system using a modified version of the Recursive Least Square method.*

MOTS-CLÉS : *Apprentissage incrémental, système d'inférence floue, reconnaissance de tracés manuscrits*

KEYWORDS: *Incremental learning, fuzzy inference system, handwriting recognition.*

1. Introduction

Avec l'émergence des assistants personnels numériques (PDA) et des téléphones mobiles de nouvelle génération (smartphones) utilisant des interfaces orientées stylo, les performances des systèmes de reconnaissance de l'écriture manuscrite en-ligne doivent être optimales. De plus en plus d'efforts sont nécessaires pour rendre ces systèmes plus robustes et adaptables afin de répondre aux nouveaux besoins des utilisateurs. Parmi ces nouveaux besoins, il apparaît intéressant de pouvoir proposer à l'utilisateur de choisir son propre groupe de gestes et de les assigner à différentes commandes interactives, par exemple : « copier », « coller », « annuler », etc. Ce contexte applicatif impose des contraintes spécifiques à la technique d'apprentissage utilisée. L'objectif est de rendre le système capable d'apprendre rapidement en utilisant peu de données. En effet, les utilisateurs sont rarement prêts à saisir chaque nouveau symbole plus d'une douzaine de fois pour l'apprentissage du système. Pour répondre à ces besoins, cette étude propose un algorithme original d'apprentissage incrémental pour les systèmes de reconnaissance de tracés manuscrits en-ligne.

La difficulté principale de cette approche est de construire « à la volée » un classifieur de tracés manuscrits, en s'appuyant sur très peu de connaissances disponibles (quelques exemples d'apprentissage), puis de l'adapter progressivement afin d'atteindre un fort taux de reconnaissance le plus rapidement possible.

Un algorithme d'apprentissage incrémental est défini dans (Polikar *et al.*, 2001) comme répondant aux critères suivants : il doit être capable d'apprendre des connaissances supplémentaires à partir des nouvelles données ; il ne doit pas nécessiter l'accès aux données d'origine (c'est-à-dire les données qui ont été utilisées pour apprendre le classifieur actuel) ; il doit préserver les connaissances déjà acquises ; et il doit être en mesure d'apprendre de nouvelles classes susceptibles d'être introduites avec de nouvelles données. Ces quatre points, qui s'appliquent pour tout problème général d'apprentissage incrémental, correspondent parfaitement aux caractéristiques particulières de notre problème, un algorithme d'apprentissage incrémental pour un système de reconnaissance auto-évolutif.

Il existe plusieurs algorithmes proposés pour effectuer l'apprentissage incrémental, bien que beaucoup d'entre eux ne répondent pas à tous les critères pour être considérés comme des approches d'apprentissage incrémental.

Nous pouvons distinguer principalement deux types d'algorithmes d'apprentissage incrémental : les algorithmes d'apprentissage incrémental de paramètres et les algorithmes d'apprentissage incrémental de structure et de paramètres. L'apprentissage incrémental de paramètres peut être assimilé à ce que l'on appelle « adaptation ». Dans ces systèmes, la structure est fixe et initialisée au départ alors que les paramètres du système sont appris progressivement en fonction des données d'apprentissage disponibles. Quelques exemples de ces systèmes sont présentés dans (Jr. *et al.*, 2007), (Jang, 1993) et (Mouchere *et al.*, 2007).

La plupart des algorithmes d'apprentissage incrémental de structure et de paramètres sont basés sur le principe de l'algorithme de clustering ART (Carpenter *et al.*, 1988), tel que (Carpenter *et al.*, 1992), (Sadri *et al.*, 2006), (Gary G. Yen, 2001) et (Lughofer, 2008). Le problème principal de ces systèmes est qu'ils sont sensibles : à la sélection du paramètre de vigilance (qui correspond au seuil de distance qui contrôle la création d'un nouveau cluster), aux niveaux de bruit dans les données d'apprentissage et à l'ordre dans lequel les données d'apprentissage sont présentées.

Une autre approche que l'on rencontre souvent en apprentissage incrémental s'appuie sur des systèmes à base d'ensemble de classifieurs, comme dans (Polikar *et al.*, 2001) et (Minku *et al.*, 2009). Dans ces systèmes, au lieu de créer de nouveaux clusters pour modéliser les nouvelles connaissances, ce sont de nouveaux classifieurs qui sont appris et ajoutés au système de façon incrémentale après l'acquisition d'une certaine quantité de données. La performance du système est donc basée sur la synergie d'un ensemble de classifieurs faibles. Cette technique peut être considérée comme un processus d'apprentissage incrémental hors-ligne (c'est-à-dire non-instantané). Elle est souvent utilisée pour apprendre incrémentalement une base de taille importante en opérant une séquence d'apprentissage « batch » sur des sous-ensembles de cette base. Cependant, les classifieurs créés sont très difficilement adaptables après leur apprentissage.

Dans un algorithme d'apprentissage incrémental, la base d'apprentissage n'est pas disponible *a priori*, puisque les exemples d'apprentissage arrivent au fil du temps. Bien que les systèmes d'apprentissage en-ligne mettent à jour et améliorent constamment leurs modèles, ils ne sont pas forcement tous basés sur une approche incrémentale. Dans certains systèmes, le modèle est complètement reconstruit à chaque étape de l'apprentissage incrémental en utilisant l'ensemble des données disponibles, ce sont des systèmes avec mémoire complète d'exemples (full instance memory) (Reinke *et al.*, 1988). Par contre, si l'algorithme d'apprentissage modifie le modèle uniquement en fonction du dernier exemple d'apprentissage, c'est un algorithme incrémental sans mémoire d'exemples (no instance memory) (Littlestone *et al.*, 1994) (Littlestone, 1991). Une troisième approche est celle des systèmes à mémoire partielle d'exemples, qui sélectionnent et gardent un sous-ensemble réduit d'exemples pour les utiliser dans les prochaines étapes d'apprentissage (Aha *et al.*, 1991).

Dans ces travaux, on constate que les modèles à base de prototypes sont généralement les mieux adaptés aux problèmes d'apprentissage incrémental rapide, où le modèle doit être modifié pour chaque nouvel exemple sans disposer d'une mémoire complète des exemples précédents.

Dans ce papier, nous proposons un nouvel algorithme d'apprentissage incrémental de structure et de paramètres avec une mémoire partielle d'exemples. Ce système est utilisé dans notre contexte pour la reconnaissance de gestes manuscrits en-ligne, et il est capable d'apprendre une nouvelle classe tout en continuant à évoluer, exemple après exemple, sans utiliser les données antérieures.

Le reste de cet article est organisé comme suit. Dans la section 2, nous présentons nos travaux antérieurs et d'autres travaux connexes qui ont inspiré le modèle proposé. Ensuite, nous décrivons dans la section 3 l'architecture du système utilisé. Les différents éléments de notre algorithme d'apprentissage sont détaillés dans la section 4. La section 5 expose les résultats des évaluations expérimentales.

2. Travaux connexes

Nous avons présenté dans nos travaux antérieurs (Almaksour *et al.,* 2008) (Almaksour *et al.,* 2009) un modèle d'apprentissage incrémental basé sur un Système d'Inférence Floue (SIF). Nous avons choisi ce système de classification à base de prototypes pour sa nature flexible qui le rend capable de s'adapter et d'évoluer selon les nouvelles données. Le SIF utilisé est du type Takagi-Sugeno d'ordre zero (Takagi *et al.,* 1985). Les règles floues font le lien entre les modèles intrinsèques (prémisses) et les sorties du système par des fonctions « conséquences ». Pour un problème à K classes, une règle R_i est construite pour chaque modèle floue P_i :

$$\textbf{SI } \vec{x} \; est \; P_i \textbf{ ALORS } y_i^1 = a_i^1 \textbf{ et ... et } y_i^k = a_i^k \qquad [1]$$

Où $\vec{x}$ est un vecteur de caractéristiques de n dimensions, a_i^c est un poids qui exprime la participation du modèle P_i dans la description de la classe C. Chaque modèle flou P_i est défini par sa fonction d'appartenance.

Pour les systèmes d'apprentissage non-incrémental, où le système est construit en utilisant une base d'apprentissage avec un nombre suffisant d'exemples, des méthodes de classification supervisée où non-supervisée sont utilisées pour créer les modèles (les centres et les matrices de covariance) pour chaque classe.

Dans (Almaksour *et al.,* 2009), nous avons proposé une stratégie de clustering incrémental pour un SIF basée sur la détection des ambiguïtés. Nous avons utilisé le rejet d'ambiguïté comme élément déclenchant pour la création d'un nouveau prototype. Nous avions pour objectif de limiter le nombre de prototypes dans le système en évitant de créer des prototypes dans des endroits où il n'y a pas de risque de confusion. Ceci élimine l'ajout des prototypes « redondants » pour une classe déjà dominante. Bien que cette méthode ait obtenu de bonnes performances, il était difficile de trouver un seuil de rejet universel et optimal. Cette méthode engendrait donc encore un trop grand nombre de prototypes pour certains problèmes, ce qui conduisait à des systèmes « lourds » parce que trop complexes.

Une autre approche de clustering incrémental a été présentée dans (Angelov *et al.,* 2004), appelée *eClustering*. Cette approche est basée sur la définition du *potentiel* d'un point, qui représente la densité associée à ce point dans l'espace des données. Un exemple à fort potentiel est considéré comme candidat pour générer un nouveau cluster. Le potentiel d'un exemple, qui a été présenté initialement dans (Yager *et al.,* 1993), peut être défini comme étant l'inverse de la somme des distances entre cet exemple et tous les autres exemples. Une formule récursive (non-itérative, en-ligne) pour le calcul

du potentiel de nouvelles données a été introduite dans (Angelov *et al.,* 2004). L'avantage principal de cette méthode de clustering incrémental est que ses paramètres (peu nombreux) sont indépendants du problème, et qu'il génère généralement beaucoup moins de prototypes que la stratégie de clustering guidée par l'ambiguïté.

Bien que la méthode eClustring génère souvent un ensemble compact de règles lorsqu'elle est utilisée pour apprendre incrémentalement un SIF, la partie intrinsèque du système est moins efficace comparée à celle générée par la stratégie guidée par l'ambiguïté. Pour faire face à ce problème, les fonctions « conséquences » du SIF doivent être améliorées en augmentant leur capacité discriminante. Pour cela, nous proposons d'utiliser un SIF de type Takagi-Sugeno d'ordre 1 en remplaçant la valeur constante dans la partie « conséquence » de la règle floue [1] par des fonctions linéaires. Plus de détails sur l'architecture du système sont présentés dans la section suivante.

3. Architecture du système

Comme mentionné précédemment, notre système est basé sur un système d'inférence floue de type Takagi-Sugeno d'ordre 1. Il se compose d'un ensemble de règles floues de la forme suivante :

$$\textbf{Règle}_i : \quad \textbf{SI } \vec{x} \; est \; P_i \; \textbf{ALORS } y_i^1 = l_i^1(\vec{x}) \; \textbf{et ... et } y_i^k = l_i^k(\vec{x}) \qquad [2]$$

où k est le nombre de classes et $l_i^m(\vec{x})$ représente les fonctions conséquences linéaires de la règle i pour la classe m :

$$l_i^m(\vec{x}) = \vec{\pi}_i^m \vec{x} = a_{i0}^m + a_{i1}^m x_1 + a_{i2}^m x_2 + ... + a_{in}^m x_n \qquad [3]$$

Pour trouver la classe de $\vec{x}$, son degré d'appartenance à chaque prototype floue $\beta_i(\vec{x})$ est calculé. Celui-ci est représenté dans notre système par une fonction à base radiale hyper-ellipsoïdale, caractérisée par un centre $\vec{\mu_i}$ et une matrice de covariance Q_i. Le degré d'activation est calculé en fonction de la distance de Mahalanobis du centre $d_{Q_i}(\vec{x}, \vec{\mu_i})$:

$$\beta_i(\vec{x}) = 1/(1 + d_{Q_i}(\vec{x}, \vec{\mu_i})) \qquad [4]$$

L'inférence floue de type somme-produit est ensuite utilisée pour calculer la sortie du système pour chaque classe :

$$y^m(\vec{x}) = \sum_{i=1}^{R} \beta_i(\vec{x}) \; l_i^m(\vec{x}) \qquad [5]$$

où R est le nombre de règles dans le système. L'étiquette de la classe gagnante est donnée en trouvant la sortie maximale et en prenant l'étiquette de classe correspondante comme réponse :

$$classe(\vec{x}) = y = argmax \; y^m(\vec{x}) \qquad m = 1, .., k \qquad [6]$$

4. Modèle d'apprentissage incrémental proposé

4.1. *Clustering incrémental*

Dans un problème d'apprentissage incrémental en-ligne, les données d'apprentissage ne sont disponibles qu'au fur et à mesure. Le système doit donc être appris en utilisant les premières données arrivées, puis continuer à évoluer de manière transparente du point de vue de l'utilisateur. Les algorithmes de clustering classiques nécessitent de connaître toutes les données afin d'effectuer le clustering. Comme mentionné dans l'introduction, un critère très important pour obtenir une solution efficace d'apprentissage incrémental consiste à éviter (ou minimiser) l'accès aux données d'apprentissage précédentes. Nous avons donc besoin d'un algorithme capable de mettre à jour les clusters chaque fois que de nouvelles données sont disponibles, sans avoir besoin de tout reconstruire, ni d'avoir accès aux données précédentes.

Quand on introduit un nouvel exemple d'apprentissage dans un mode d'apprentissage en-ligne, il pourra, soit renforcer l'information contenue dans les données précédentes et représentée par les clusters existants, soit apporter une nouvelle information suffisamment importante pour former un nouveau cluster ou modifier un cluster existant. L'importance d'un exemple dans le processus de clustering est évaluée par sa valeur de *potentiel*. Le potentiel d'un exemple est défini comme étant l'inverse de la somme des distances entre un exemple et tous les autres exemples (Yager *et al.*, 1993) :

$$Pot(x(t)) = \frac{1}{1 + \sum_{i=1}^{t-1} \|x(t) - x(i)\|^2} \quad [7]$$

Une méthode récursive pour le calcul du potentiel d'un nouvel exemple a été introduit dans (Angelov *et al.*, 2004). Cela constitue une solution prometteuse pour tout problème de clustering incrémental. La formule récursive évite la mémorisation de l'ensemble des données précédentes, mais conserve - en utilisant quelques variables - la distribution de densité dans l'espace. Le potentiel est défini de la façon suivante :

$$Pot(x(t)) = \frac{t-1}{(t-1)\alpha(t) + \gamma(t) - 2\zeta(t) + t - 1} \quad [8]$$

où

$$\alpha(t) = \sum_{j=1}^{n} x_j^2(t) \quad [9]$$

$$\gamma(t) = \gamma(t-1) + \alpha(t-1), \quad \gamma(1) = 0 \quad [10]$$

$$\zeta(t) = \sum_{j=1}^{n} x_j(t)\eta_j(t), \quad \eta_j(t) = \eta_j(t-1) + x_j(t-1), \quad \eta_j(1) = 0 \quad [11]$$

L'introduction d'un nouvel exemple affecte les valeurs du potentiel des centres des clusters existants, qui peuvent être mis à jour récursivement par l'équation suivante :

$$Pot_t(\mu_i) = \frac{(t-1)Pot_{t-1}(\mu_i)}{t - 2 + Pot_{t-1}(\mu_i) + Pot_{t-1}(\mu_i) \sum_{j=1}^{n} \|\mu_i - x(t-1)\|_j^2} \quad [12]$$

Si le potentiel du nouvel exemple est plus élevé que le potentiel des centres existants, alors cet exemple sera le centre d'un nouveau cluster (et une nouvelle règle floue sera ajoutée dans le cas de notre SIF). Si l'exemple à fort potentiel est proche d'un centre existant μ_i (la distance euclidienne entre x(t) et u_i est inferieure à $\epsilon), alors ille remplace et aucun cluster supplmentaire ne sera cr. Bien que la valeur de \epsilon$ contrôle le nombre de règles crées dans le système, elle n'a pas beaucoup d'effet sur la performance. Nous avons trouvé expérimentalement qu'une valeur de 0.1 peut être un bon choix pour la plupart des problèmes (à condition que les valeurs d'entrée soit entre 0 et 1).

4.2. *Apprentissage incrémental des paramètres des fonctions conséquences linéaires*

Le problème de l'apprentissage des fonctions conséquences dans un SIF se résume à résoudre un système d'équations linéaires exprimées par :

$$\Psi_i \, \Pi \; = \; Y_i \qquad i = 1, 2, ..., t \tag{13}$$

où Π est la matrice de tous les paramètres des conséquences linéaires du système :

$$\Pi = \begin{bmatrix} \vec{\pi}_1^1 & \vec{\pi}_1^2 & ... & \vec{\pi}_1^m \\ \vec{\pi}_2^1 & \vec{\pi}_2^2 & ... & \vec{\pi}_2^m \\ ... & ... & ... & ... \\ \vec{\pi}_r^1 & \vec{\pi}_r^2 & ... & \vec{\pi}_r^m \end{bmatrix}$$

m représente le nombre de classes, r est le nombre de règles floues,
$\Psi_i \; = \; [\beta_1(\vec{x}_i)\vec{x}_i, \beta_2(\vec{x}_i)\vec{x}_i, ..., \beta_r(\vec{x}_i)\vec{x}_i]$ est le vecteur d'entrées pondérées par les degrés d'activation des prototypes, et Y_i est la sortie objectif.

Résoudre ce système avec la méthode des moindres carrés consiste à minimiser la fonction de coût suivante :

$$E = \sum_{i=1}^{t} \left\| \Psi_i \, \Pi \; - \; Y_i \right\|^2 \tag{14}$$

Dans les problèmes de régression linéaire des modèles physiques et industriels, un bruit peut à tout moment s'ajouter sur le signal d'entrée. Afin de stabiliser la résolution du système et de lisser la solution trouvée, un terme de régularisation (connu comme la régularisation Tychonoff) est ajouté au système. Le système est donc exprimé comme suit :

$$(\Psi_i \; + \delta I)\Pi \; = \; Y_i \qquad i = 1, 2, ..., t \tag{15}$$

On réécrit donc la fonction de coût :

$$E = \sum_{i=1}^{t} \left\| \Psi_i \, \Pi \; - \; Y_i \right\|^2 + \omega \left\| \Pi \right\|^2 \tag{16}$$

où $\omega = \delta^2$ est un nombre positif appelé le paramètre de régularisation, et I est la matrice identité.

Le paramètre de régularisation est important dans notre SIF parce que l'adaptation de prémisses engendre un bruit sur l'entrée Ψ (Nous pouvons obtenir $\Psi_{t1} \neq \Psi_{t2}$ pour $\vec{x}_{t1} = \vec{x}_{t2}$ à cause de l'évolution de prémisses). La solution qui minimise la fonction de coût de l'équation 16 est la suivante :

$$\Pi_t = (\sum_{i=1}^{t} \Psi_i \Psi_i^T + \omega I)^{-1} \; . \; \sum_{i=1}^{t} \Psi_i Y_i \qquad [17]$$

Nous réécrivons l'équation 17 en remplaçant $(\sum_{i=1}^{t} \Psi_i \Psi_i^T + \omega I)$ et $(\sum_{i=1}^{t} \Psi_i Y_i)$ par Φ_t et Z_t, respectivement :

$$\Pi_t = \Phi_t^{-1} \; . Z_t \qquad [18]$$

En isolant le terme correspondant à $i = t$, nous pouvons réécrire Φ_t :

$$\Phi_t = \left[\sum_{i=1}^{t-1} \Psi_i \Psi_i^T + \omega I \right] + \Psi_t \Psi_t^T \qquad [19]$$

Ainsi, nous pouvons mettre à jour la matrice Φ avec la formule récursive suivante :

$$\Phi_t = \Phi_{t-1} + \Psi_t \Psi_t^T \qquad [20]$$

De la même façon, nous pouvons déduire une formule récursive pour la mise à jour de la matrice Z :

$$Z_t = Z_{t-1} + \Psi_t Y_t \qquad [21]$$

Afin de calculer Π_t par l'équation 18, nous avons besoin de calculer l'inverse de la matrice Φ. Dans la pratique, nous essayons généralement d'éviter ce genre d'opérations car elles nécessitent beaucoup de temps de calcul. De plus, nous préférons avoir également une formule récursive pour calculer Π de façon incrémental et efficace. Nous pouvons réaliser ces deux objectifs grâce au lemme d'inversion de matrice suivant :

Lemme 1 : Soit $A = B^{-1} + CD^{-1}C^T$, nous pouvons exprimer l'inverse de A comme suit :

$$A^{-1} = B - BC(D + C^T BC)^{-1} C^T B \qquad [22]$$

Afin d'appliquer Lemme 1 sur l'équation 20, nous faisons les substitutions suivantes :

$$A = \Phi_t, B^{-1} = \Phi_{t-1}, C = \Psi_t, D = 1$$

Ainsi, nous pouvons obtenir la formule récursive de l'inverse de la matrice Φ :

$$\Phi_t^{-1} = \Phi_{t-1}^{-1} - \frac{\Phi_{t-1}^{-1} \; \Psi_t \; \Psi_t^T \; \Phi_{t-1}^{-1}}{1 + \Psi_t^T \; \Phi_{t-1}^{-1} \; \Psi_t} \qquad [23]$$

Puis, et dans le but d'obtenir une équation récursive pour calculer Π, nous écrivons :

$$\begin{aligned}\Pi_t &= \Phi_t^{-1} Z_t = \Phi_t^{-1}(Z_{t-1} + \Psi_t Y_t) = \Phi_t^{-1}(\Phi_{t-1}\Pi_{t-1} + \Psi_t Y_t) \\ &= \Phi_t^{-1}((\Phi_t - \Psi_t \Psi_t^T)\Pi_{t-1} + \Psi_t Y_t) = \Pi_{t-1} - \Phi_t^{-1}\Psi_t\Psi_t^T\Pi_{t-1} + \Phi_t^{-1}\Psi_t Y_t \\ &= \Pi_{t-1} - \Phi_t^{-1}\ \Psi_t\ (Y_t - \Psi_t^T\ \Pi_{t-1}) \qquad [24]\end{aligned}$$

L'initialisation de l'algorithme consiste à déterminer deux quantités :

– Π_0 : Dans la pratique, et quand aucune pré-connaissance n'est disponible, Π_0 est initialisé à 0.

– Φ_0^{-1} : Etant donné $\Phi_t = \sum_{i=1}^{t} \Psi_i \Psi_i^T + \omega I$ et en mettant t à 0, nous trouvons que $\Phi_0^{-1} = \omega^{-1} I$, où ω est le paramètre de régularisation.

Des grandes valeurs de ω^{-1} (entre 10^2 et 10^4) sont généralement adoptées quand le taux de bruit sur le signal d'entrée est important, ce qui est le cas dans notre SIF surtout au début de l'apprentissage où des modifications importantes sont effectuées sur les prototypes. L'effet de la valeur de ω^{-1} sur la performance de l'algorithme en fonction du niveau de bruit d'entrée est étudié en détail dans (Moustakides, 1997).

Quand une nouvelle règle est ajoutée au système, ses paramètres sont initialisés par la moyenne pondérée des paramètres des autres règles :

$$\Pi_t = \begin{bmatrix} \vec{\pi}^1_{1(t-1)} & \vec{\pi}^2_{1(t-1)} & \cdots & \vec{\pi}^m_{1(t-1)} \\ \vec{\pi}^1_{2(t-1)} & \vec{\pi}^2_{2(t-1)} & \cdots & \vec{\pi}^m_{2(t-1)} \\ \cdots & \cdots & \cdots & \cdots \\ \vec{\pi}^1_{r(t-1)} & \vec{\pi}^2_{r(t-1)} & \cdots & \vec{\pi}^m_{r(t-1)} \\ \vec{\pi}^1_{(r+1)t} & \vec{\pi}^2_{(r+1)t} & \cdots & \vec{\pi}^m_{(r+1)} \end{bmatrix} \qquad [25]$$

où

$$\vec{\pi}^c_{(r+1)t} = \sum_{i=1}^{r} \beta_i(\vec{x}_t)\vec{\pi}^c_{i(t-1)} \qquad [26]$$

En outre, la matrice Φ^{-1} est étendue de la façon suivante :

$$\Phi_t^{-1} = \begin{bmatrix} \rho \begin{bmatrix} \Phi_{t-1}^{-1} \end{bmatrix} & \begin{bmatrix} 0 \end{bmatrix} \\ \begin{bmatrix} 0 \end{bmatrix} & \begin{bmatrix} \Omega^{-1} & \cdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \cdots & \Omega^{-1} \end{bmatrix} \end{bmatrix} \qquad [27]$$

où $\rho = (r^2+1)/r^2$.

4.3. *Adaptation des prémisses*

Comme on peut le noter dans la section 4.1, la condition permettant d'avoir un fort potentiel est très dure, et elle est inversement proportionnelle au nombre croissant de

données. Ainsi, on peut imaginer que le centre d'un cluster $\vec{\mu_i}$ qui n'est pas vraiment en position optimale étant donné l'historique des données présentées, ne sera pas modifié car il conserve la plus grande valeur de potentiel par rapport aux autres données. Par conséquent, le processus de clustering incrémental de la partie prémisse d'un SIF ne sera pas en mesure de profiter des données qui n'ont pas un potentiel suffisamment élevé pour déplacer (ou déformer) les clusters existants.

Nous améliorons le processus de clustering incrémental (décrites dans la section 4.1) par un algorithme d'adaptation qui permet de modifier tous les modèles du SIF en les recentrant et les déformant pour chaque nouvelle donnée. Notre algorithme est inspiré par la méthode FLVQ (Fuzzy Learning Vector Quantization) (Chung *et al.*, 1994). Dans cette méthode, plus l'activation β_i de la prémisse de la règle i est loin de sa valeur objectif β_i^*, plus le prototype doit être déplacé :

$$\vec{\Delta\mu_i} = \lambda * (\beta_i^* - \frac{\beta_i(\vec{x})}{\sum_{q=1}^{r} \beta_q(\vec{x})}) * (\vec{x} - \vec{\mu_i}) \quad [28]$$

où le paramètre d'adaptation λ est compris entre 0 et 1. Le score objectif β_i^* vaut 1 si l'exemple $\vec{x}$ et le prototype P_r sont de la même classe et 0 sinon. L'inconvénient principal d'utiliser directement cette méthode dans notre SIF est qu'elle ne prend en considération que le score objectif de ce prototype et pas sa participation dans le score final en sortie. Par conséquent, nous avons étendu cet algorithme pour qu'il modifie tous les prototypes du système pour chaque nouvel exemple en fonction de leurs participations effectives au processus de reconnaissance (et non en fonction de leur score objectif au niveau des fonctions d'activation). Ainsi, la mise à jour d'un prototype doit améliorer le score de chaque classe, donc, le déplacement $\vec{\Delta\mu_i}$ doit être d'autant plus important que le score de la classe y^c est différent de sa valeur objectif $\hat{y}^c$; plus la participation du prototype au score final de la classe y_i^c est élevée et plus la prémisse de la règle β_i sera fortement activée :

$$\begin{aligned} \vec{\Delta\mu_i} &= \lambda * \left(\beta_i(\vec{x}) \sum_{c=1}^{m} ((\hat{y}^c(\vec{x}) - y^c(\vec{x}))\ y_i^c(\vec{x})) \right) * (\vec{x} - \vec{\mu_i}) \\ &= \lambda * \left(\beta_i^2(\vec{x}) \sum_{c=1}^{m} ((\hat{y}^c(\vec{x}) - y^c(\vec{x}))\ \vec{l_i^c}(\vec{x})) \right) * (\vec{x} - \vec{\mu_i}) \end{aligned} \quad [29]$$

où $\hat{y}^c$ vaut 1 si c est la classe de $\vec{x}$ et 0 sinon.

Comme mentionné dans la section 4.1, lorsque le potentiel du nouvel exemple est plus élevé que les potentiels des centres existants alors cet exemple sera le centre d'un nouveau prototype P_{r+1}, et la matrice de covariance Q_{R+1}, qui définit la zone d'influence (selon la distance de Mahalanobis), sera initialisée par une matrice proportionnelle à la matrice identité (ce qui se traduit par une forme hyper-sphérique).

Afin de mieux représenter la répartition du nuage de données qui ont incrémentalement formé le prototype, la forme du prototype (donnée par sa matrice de covariance) doit être adaptée et modifiée en fonction de chaque nouvel exemple. Une formule

récursive pour mettre à jour l'inverse de la matrice de covariance dans un contexte non-supervisé est donnée dans (De Backer *et al.*, 2001). Elle utilise les activations des prototypes dans le calcul de la matrice de covariance. Dans (Mouchere *et al.*, 2007), une nouvelle version de cette formule est présentée en utilisant, en plus des activations des prototypes, la participation du prototype à la reconnaissance de chaque classe, et l'erreur commise sur chaque classe. Sur la base de la formule présentée dans (Mouchere *et al.*, 2007), et en l'adaptant à notre architecture (SIF d'ordre 1), nous obtenons la formule récursive suivante pour mettre à jour l'inverse de la matrice de covariance Q_i :

$$Q_i^{-1} = \frac{Q_i^{-1}}{1-\alpha\delta_i} - \frac{\alpha\delta_i}{1-\alpha\delta_i} \cdot \frac{(Q_i^{-1}\vec{d}) \cdot (Q_i^{-1}\vec{d})^T}{1+\alpha\delta_i(\vec{d}^T Q_i^{-1}\vec{d})} \quad [30]$$

$$\delta_i = \beta_i^2(\vec{x}) \sum_{c=1}^{m} ((\hat{y}^c(\vec{x}) - y^c(\vec{x}))\ \vec{l}_i^c(\vec{x})) \quad [31]$$

avec $\vec{d} = \vec{x} - \vec{\mu_r}$ et α un facteur d'apprentissage.

Les paramètres λ et α peuvent être remplacés pour les rendre indépendants du problème par $\frac{1}{s_i+1}$ où s_i représente le nombre de mises à jour qui ont déjà été appliquées sur ce prototype. Cela permet au prototype de s'adapter plus vite au début, puis de se stabiliser avec le temps.

4.4. *Stabilité de l'apprentissage des paramètres des fonctions conséquences*

L'une des propriétés favorables de la méthode des moindres carrés récursifs pondérés, est qu'elle converge vers la solution optimale pour chaque étape de l'apprentissage. Mais, cette propriété n'est vraie que lorsque les poids (les activations des prémisses dans notre cas) qui ont été associés aux exemples précédents restent inchangés. Dans notre système, les prototypes flous qui forment la partie prémisse sont dynamiques : ils peuvent être recentrés, déplacés ou déformés par la méthode de clustering incrémental (section 4.1) et l'algorithme d'adaptation (section 4.3). Par conséquent, les activations plus anciennes qui ont participé à l'apprentissage des paramètres des conséquences pour les données précédentes ne sont plus les mêmes. Cela rend l'estimation antérieure de ces paramètres incohérente avec le modèle actuel de prémisse.

Dans (Lughofer, 2008), l'auteur aborde ce problème de stabilité. Il propose l'idée d'introduire, après chaque modification de prémisse, un vecteur de correction pour les paramètres linéaires et une matrice de correction pour leur matrice de covariance, afin d'aller vers la solution optimale en fonction du degré de changement dans la partie prémisse. A notre connaissance, cette idée n'a jamais été formalisée étant donnée la complexité d'estimation de ces vecteurs et matrices de corrections. Ce problème complexe reste donc aujourd'hui toujours ouvert.

Pour faire face à ce problème de stabilité, nous proposons d'appliquer le clustering incrémental et l'algorithme d'adaptation sur la partie prémisse en mode « batch », au lieu de l'appliquer à chaque exemple. De cette façon, nous continuons à appliquer l'algorithme de la mise à jour des paramètres de conséquences pour chaque exemple avec une structure temporairement fixe de prémisse. Pour cela, nous gardons les exemples de données dans une mémoire tampon de taille F. Ainsi, après l'introduction de F exemples de données, le clustering incrémental et l'algorithme d'adaptation sont appliqués pour chaque exemple de la mémoire tampon. Puis, un réajustement des paramètres des conséquences est fait en appliquant la méthode MCRP sur les exemples de la mémoire (avec la structure modifiée de prémisse). Après ce processus de réajustement, la mémoire tampon est vidée et la structure des prémisses est gelée, alors que le processus de la mise à jour des paramètres des conséquences se poursuit pour chaque nouvel exemple.

L'algorithme complet de notre modèle d'apprentissage incrémental est résumé dans Algorithme 1.

5. Synthèse de données artificielles

Afin d'obtenir un apprentissage incrémental rapide et de surmonter le manque de données disponibles au début du processus d'apprentissage des nouvelles classes de gestes, nous choisissons de synthétiser des exemples d'apprentissage artificielles en déformant de façon réaliste et naturelle les exemples de gestes disponibles. Ces données d'apprentissage supplémentaires vont accélérer l'apprentissage en ayant un impact d'une part sur l'adaptation des prémisses et particulièrement sur l'estimation des matrices de covariance des prototypes, et d'autre part sur l'estimation des paramètres des fonctions conséquences linéaires. Comme il est proposé dans (Mouchere *et al.*, 2006), différentes techniques de génération des gestes manuscrits en ligne peuvent être utilisées pour générer des exemples artificiels, en restant fidèle au geste réel. Dans (Mouchere *et al.*, 2006), deux stratégies pour générer des gestes manuscrits ont été proposées. La première utilise les distorsions classiques d'image (de type hors-ligne), comme l'étirement et l'inclinaison. La seconde, fondée sur la particularité de l'écriture manuscrite en-ligne, applique deux distorsions en-ligne sur le geste : modification de la vitesse et modification de la courbure. Il s'agit de générer plusieurs variations d'un même geste à partir d'un seul exemple de ce geste en-ligne. Pour cela nous appliquons successivement, sur l'exemple disponible, une ou plusieurs déformations. À chaque nouvelle génération il faut choisir des paramètres de déformation différents pour obtenir une certaine variabilité dans les données générées. Mais il ne faut pas non plus utiliser des paramètres engendrant des déformations trop importantes pour ne pas synthétiser un geste qui ne ressemble plus à celui d'origine. Pour chaque paramètre, une borne minimale et une borne maximale sont fixées pour limiter les déformations générées. Ces bornes sont fixées a priori et empiriquement en visualisant pour chacune d'elles les déformations moyennes obtenues pour plusieurs classes de gestes. Une fois fixées, les bornes restent les mêmes tout au long des expérimentions.

Algorithme 1 : Algorithme d'apprentissage incrémental de SIF d'ordre 1

Initialisation :
$\epsilon = 0.1$
$\omega^{-1} = 1000$
pour chaque *nouvel exemple* $\vec{x}$ **faire**
  si $\vec{x}$ *est le premier exemple d'une nouvelle classe* **alors**
    créer un nouveau prototype autour de $\vec{x}$;
    initialiser son potentiel par 1;
    ajouter une nouvelle règle floue au système;
    étendre la matrice des paramètres des conséquences comme dans [25] et [26];
    mettre à jour et étendre la matrice de covariance comme dans [27];
  sinon
    calculer les activations des règles floues par [4];
    déterminer la classe de $\vec{x}$ par [6];
    obtenir la classe effective de $\vec{x}$;
    calculer le potentiel de $\vec{x}$ par [8];
    mettre à jour les potentiels des centres des prototypes existants en utilisant [12];
    si $Pot(\vec{x}) > Pot_k(\vec{\mu}_i) \ \ \forall i \in [1, R]$ **alors**
      si $d(\vec{x}, \vec{\mu}_i) < \epsilon$ **alors**
        mettre $\vec{x}$ le nouveau centre du prototype P_i et garder les mêmes conséquences que la *règle*$_i$;
      sinon
        créer un nouveau prototype autour de $\vec{x}$;
        initialiser son potentiel par 1;
        ajouter une nouvelle règle floue au système;
        étendre la matrice des paramètres des conséquences comme dans [25] et [26];
        mettre à jour et étendre la matrice de covariance comme dans [27];
      fin
    fin
    mettre à jour les paramètres des conséquences par [23] et [24];
    ajouter $\vec{x}$ à la mémoire tampon;
    si *mémoire tampon est pleine* **alors**
      pour chaque *exemple* $\vec{x}_f$ *dans la mémoire tampon* **faire**
        Appliquer l'adaptation des prémisses en fonction de $\vec{x}_f$ par [29] et [30];
      fin
      pour chaque *exemple* $\vec{x}_f$ *dans la mémoire tampon* **faire**
        mettre à jour les paramètres des conséquences en fonction de $\vec{x}_f$ par [23] et [24];
      fin
      vider la mémoire tampon;
    fin
  fin
fin

6. Evaluation

Dans cette section, nous allons analyser la performance et le comportement de notre modèle d'apprentissage incrémental sur deux types d'expérience. La première expérience est menée sur un problème d'apprentissage incrémental spécifique lié à notre contexte applicatif, la reconnaissance des gestes manuscrits. Cette expérience permet à la fois d'évaluer la performance du model d'apprentissage incrémental présenté, mais aussi l'impact de l'utilisation des donnés d'apprentissage synthétiques sur cette performance. Dans la deuxième partie de cette étude expérimentale, nous testons le modèle d'apprentissage incrémental présenté sur des benchmarks connus de classification, ce qui permet de comparer nos résultats avec ceux de méthodes connues d'apprentissage non-incrémental, tout en offrant la possibilité à d'autres approches incrémentales de se comparer à notre méthode.

6.1. *Première expérience : reconnaissance de gestes manuscrits*

Nous avons mené cette expérience sur un ensemble de données de 25 gestes manuscrits en-ligne (figure 1). Chaque geste est décrit par un ensemble de 10 caractéristiques. Nous exploitons 10 tests différents sur 10 bases mono-scripteurs saisies par des personnes différentes sur un Tablet PC. Cette base, fruit de la collaboration de l'équipe IMADOC de l'IRISA et du laboratoire Synchromedia de l'ETS, est disponible gratuitement[1]. Les résultats présentés sont la moyenne de 10 essais. Chaque scripteur a saisi 100 exemples de chaque symbole, soit 2500 symboles dans chaque base mono-scripteur. Nous utilisons la moitié de chaque base pour le processus d'apprentissage incrémental, et le reste pour estimer l'évolution de la performance du système pendant le processus d'apprentissage. Afin d'obtenir des résultats non-biaisés par l'effet de l'ordre de l'introduction des données d'apprentissage, nous répétons l'expérience pour chaque scripteur 30 fois avec un ordre aléatoire différent à chaque fois, et puis nous calculons le résultat moyen.

Trois modèles d'apprentissage incrémental sont comparés dans cette expérience :

– Modèle I : SIF d'ordre 1 avec un clustering incrémental à base de potentiel et la méthode MCRP. Ce modèle sera le modèle de référence et il est utilisé pour mettre en relief les deux principales contributions de ce papier (Modèle II et III).

– Modèle II : modèle I + adaptation des prémisses.

– Modèle III : modèle II + utilisation de la synthèse d'exemples artificiels.

Les paramètres Ω et F sont fixés à 1000 et 30 respectivement. Quand la synthèse d'exemples artificiels est utilisée, 30 exemples synthétiques sont générés à partir de chaque exemple réel.

Dans le premier protocole expérimental, nous introduisons les 25 classes de gestes dès le début du processus d'apprentissage. La figure 2 montre l'évolution du taux

1. http ://www.synchromedia.ca/web/ets/gesturedataset

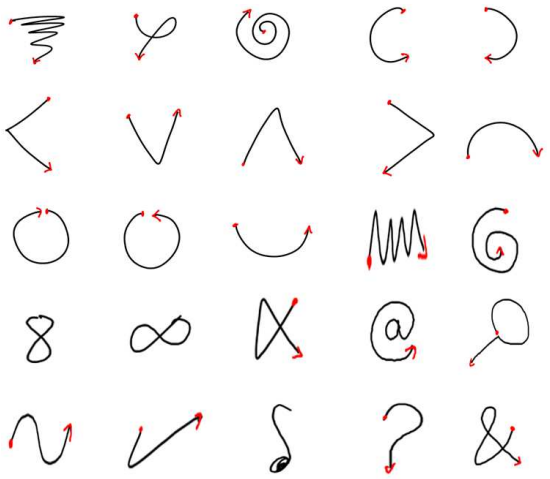

Figure 1. *Les gestes manuscrits utilisés*

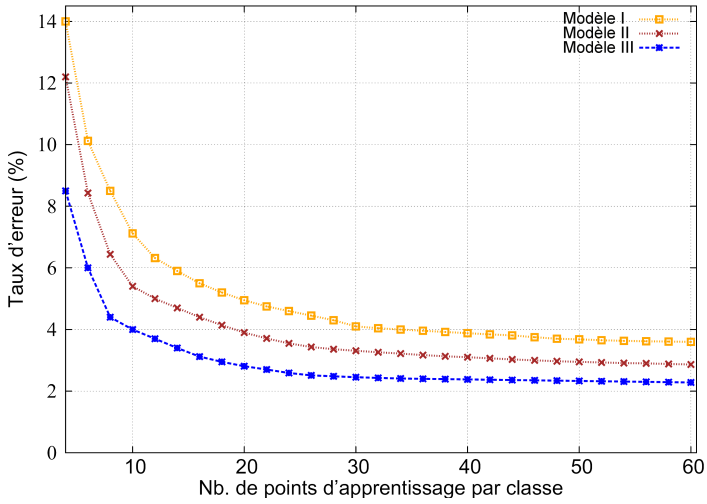

Figure 2. *Evolution du taux de reconnaissance lors du processus d'apprentissage incrémental*

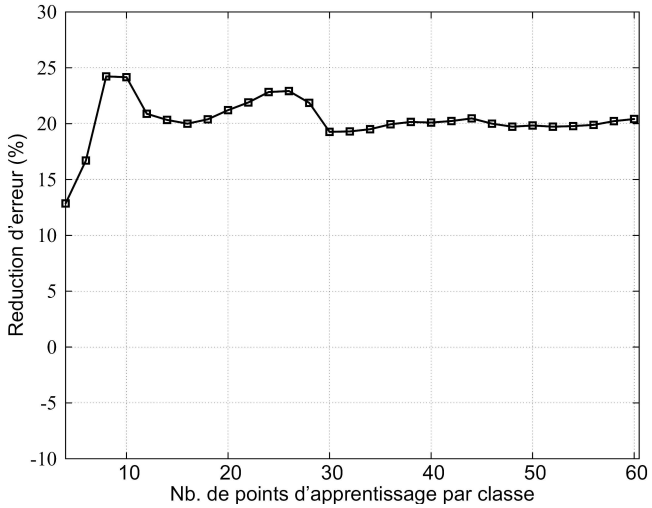

Figure 3. *Réduction d'erreur grâce à l'adaptation des prémisses (modèle II)*

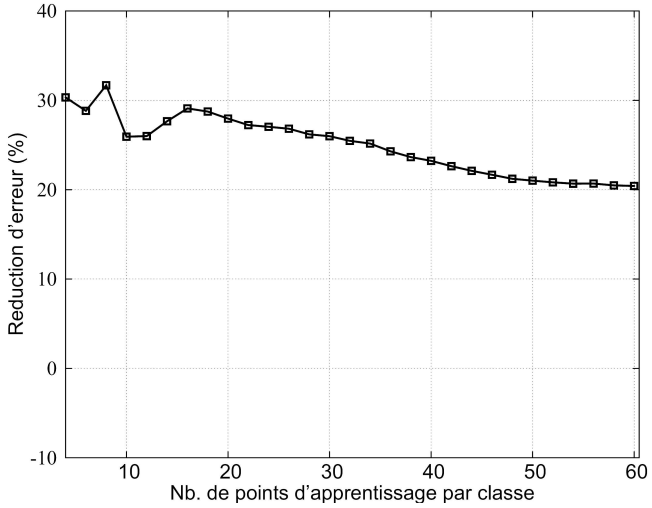

Figure 4. *Réduction d'erreur grâce à la synthèse (modèle III)*

d'erreur sur la base de test lors du processus d'apprentissage incrémental pour les trois modèles. De plus, on montre via les figures 3 et 4 la réduction relative du taux d'erreur acquise grâce à l'adaptation des prémisses (Modèle II) et grâce à la synthèse des données artificielles (Modèle III), respectivement. Nous remarquons que l'adaptation des prémisses permet au modèle II de dépasser le modèle I et de réduire le taux d'erreur de 20% à 25%. Nous pouvons également remarquer que la synthèse de données artificielles réduit le taux d'erreur de façon significative (entre 30% et 20% environ), notamment au début du processus d'apprentissage quand seulement très peu d'exemples d'apprentissage sont disponibles.

Dans le deuxième protocole, on simule un contexte d'application réelle dans lequel l'apprentissage commence par un ensemble de classes de gestes, puis l'utilisateur ajoute progressivement d'autres classes de gestes en fonction de ses besoins. Nous voulons mesurer dans cette expérience le comportement du modèle face au besoin d'apprendre à la volée de nouvelles classes. Pour cela, nous découpons l'ensemble des classes des gestes en trois sous-ensembles A,B et C avec 10,7 et 8 classes respectivement. Le processus d'apprentissage commence en introduisant des exemples d'apprentissage du sous-ensemble A, et en évaluant la performance sur un sous-ensemble de la base de test qui contient les classes de A. Ensuite, nous ajoutons en deux temps les classes de B puis les classes de C, en élargissant la base de test à chaque fois. Les figures 5, 6 et 7 montrent les résultats des tests de ce protocole expérimental. Nous remarquons notamment sur la figure 7 le rôle important de la synthèse après l'ajout des nouvelles classes au système. La synthèse couplée à l'adaptation (Modèle III), permet une accélération et une optimisation de l'apprentissage incrémental lors de l'ajout à la volée de nouvelles classes (réduction d'erreurs entre 20% et 40%).

6.2. *Deuxième expérience : benchmark classique (PenDigits)*

Nous avons utilisé pour cette expérience, la base « Pen-Based Recognition of Handwritten Digits » de UCI-repository (Asuncion *et al.*, 2007). Cette base contient 7494 exemples d'apprentissage et 3498 exemples de test. Chaque exemple est représenté par 16 caractéristiques. Elle contient 10 classes pour les 10 chiffres. D'après (Lughofer *et al.*, 2007), le taux d'erreur minimal réalisé sur cette base par un classifieur de type K-nn (avec une valeur de k optimisée empiriquement) est de 2.6%. D'une façon similaire au deuxième protocole de la première expérience, nous introduisons les classes des chiffres en deux temps. L'apprentissage commence par les chiffres de 0 à 6, puis nous ajoutons dans un second temps les chiffres de 7 à 9 au système de classification. Nous pouvons remarquer sur les figures 8 et 9 la supériorité en performance du modèle II par rapport au modèle I. Nous pouvons également remarquer que la performance du modèle d'apprentissage incrémental proposé peut atteindre ou dépasser les performances des modèles d'apprentissage classique connus. Nous remarquons également la stabilité du processus d'apprentissage du modèle II puisque l'on constate qu'il n'y a quasi aucune chute de performance lors de l'introduction à la volée de nouvelles classes.

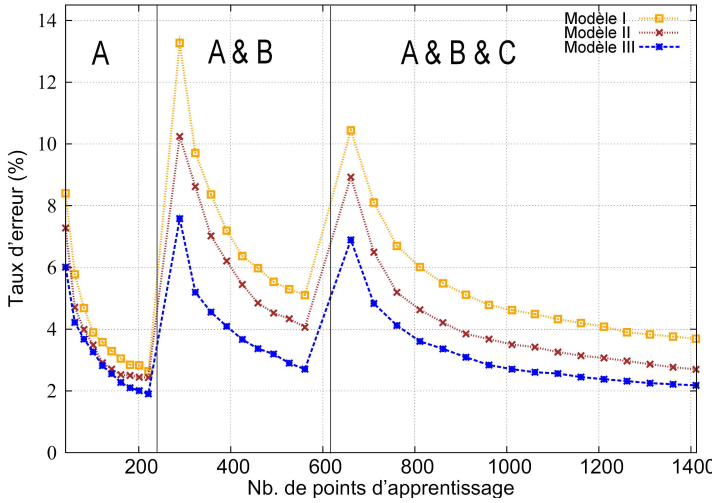

Figure 5. *Evolution du taux de reconnaissance et stabilité lors de l'introduction des nouvelles classes*

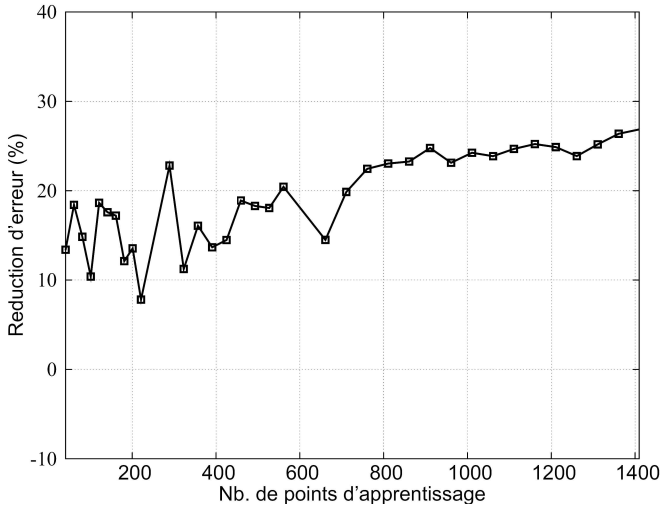

Figure 6. *Réduction d'erreur grâce à l'adaptation des prémisses (modèle II)*

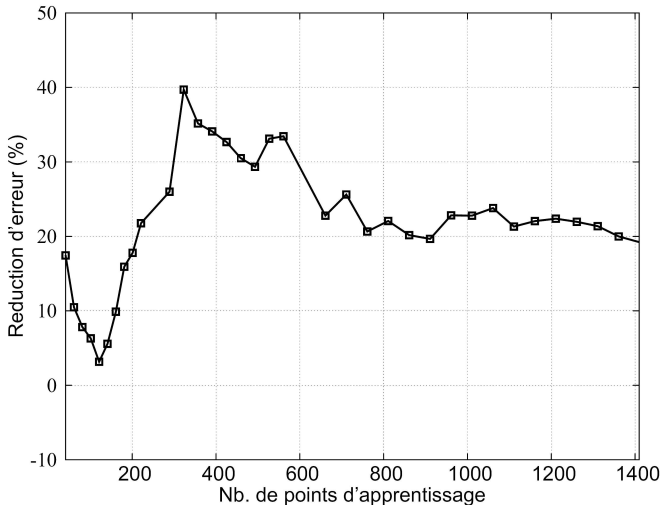

Figure 7. *Réduction d'erreur grâce à la synthèse (modèle III)*

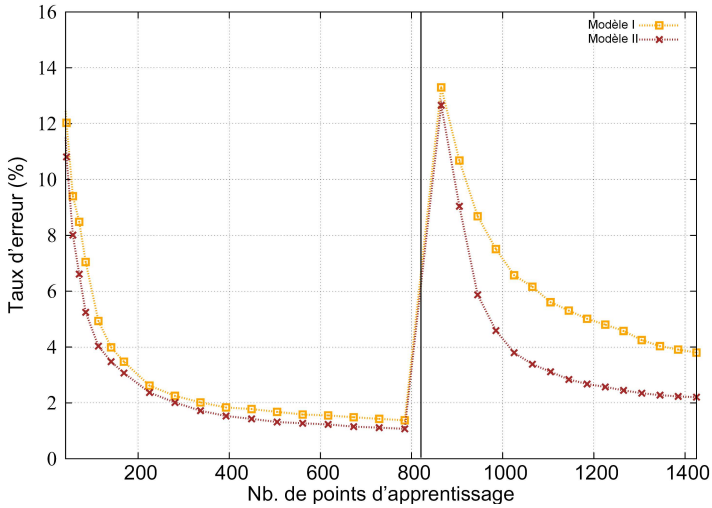

Figure 8. *Evolution du taux de reconnaissance et stabilité lors de l'introduction des nouvelles classes (PenDigits)*

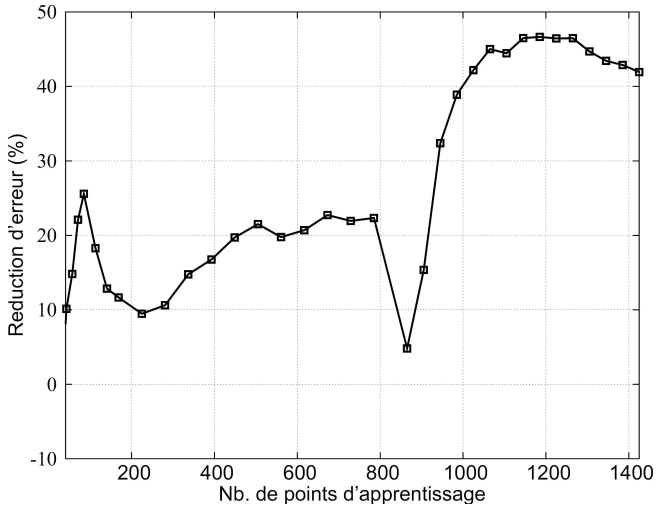

Figure 9. *Réduction d'erreur grâce à l'adaptation des prémisses (PenDigits)*

6.3. *Troisième expérience : benchmark classique (Vehicle)*

Nous avons mené la troisième expérience sur la base « Vehicle silhouettes » de UCI-repository (Asuncion *et al.*, 2007) (Siebert, 1987). Il s'agit de classer une silhouette donnée comme l'un des quatre types de véhicules, en utilisant un ensemble de caractéristiques extraites de la silhouette. La base contient 846 exemples de 4 classes de véhicules. Chaque exemple est représenté par 18 caractéristiques. Nous commençons l'apprentissage par deux classes de véhicules, puis nous ajoutons la troisième puis la quatrième. La base « Vehicle silhouettes » a été utilisée dans (Chen *et al.*, 2010) et le taux d'erreur minimal réalisé en utilisant les arbres de décision est de 26%. Avec des classifieurs de type MLP et K-nn (avec k=5), le taux d'erreur est de 20% et de 20.5% respectivement. Nous remarquons sur les figures 10 et 11 que l'adaptation des prémisses réduit le taux d'erreur d'environ 25%. Nous pouvons aussi remarquer que nous obtenons les mêmes améliorations grâce au Modèle II sur cette troisième expérience ce qui confirme les bonnes propriétés du modèle proposé sur des problèmes de natures très différentes.

7. Conclusion

Dans le contexte des systèmes de reconnaissance de gestes manuscrits, nous avons présenté dans ce papier un nouvel algorithme d'apprentissage incrémental basé sur un système d'inférence floue d'ordre 1. Grâce à cet algorithme, le système de reconnaissance peut commencer à apprendre à partir de zéro et avec peu de données d'apprentissage. Il peut aussi s'améliorer et s'adapter aux données nouvellement disponibles.

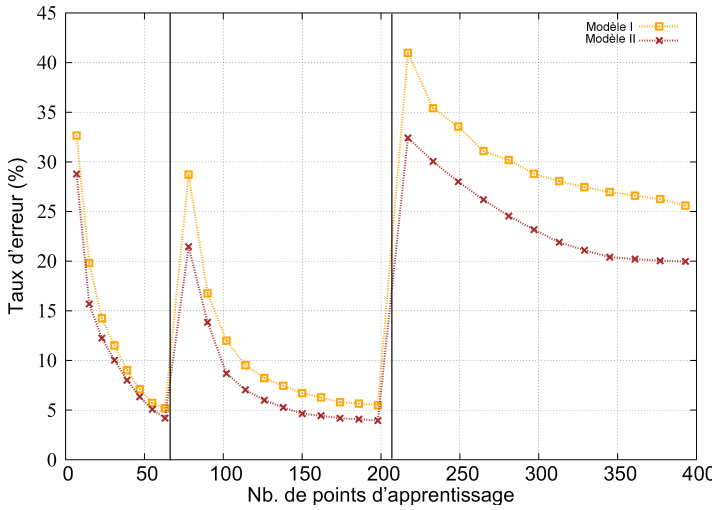

Figure 10. *Evolution du taux de reconnaissance et stabilité lors de l'introduction des nouvelles classes (Vehicle)*

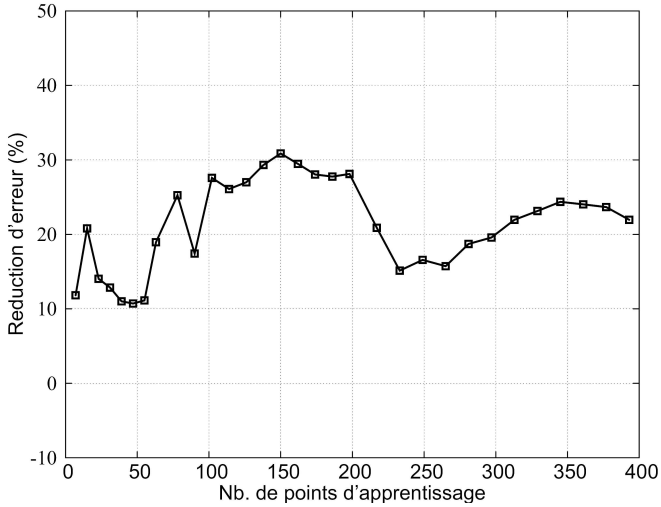

Figure 11. *Réduction d'erreur grâce à l'adaptation des prémisses (Vehicle)*

Dans le domaine de la reconnaissance de gestes manuscrits, nous avons couplé des techniques de synthèse d'exemples artificiels à partir de gestes manuscrits originaux, afin d'accélérer le processus d'apprentissage lorsque très peu de données sont disponibles. Pour les travaux futurs, un accent particulier sera mis sur la recherche d'une estimation d'un vecteur de correction des conséquences linéaires après chaque adaptation des prémisses, ce qui peut améliorer la performance du système et éliminer le besoin d'une mémoire tampon.

Remerciements

Nous remercions la Région Bretagne pour sa contribution au financement de ces recherches dans le cadre du projet ScriptEverywhere

8. Bibliographie

Aha D. W., Kibler D., Albert M. K., « Instance-Based Learning Algorithms », *Mach. Learn.*, vol. 6, n° 1, p. 37-66, 1991.

Almaksour A., Anquetil E., « Fast Incremental Learning Strategy Driven by Confusion Reject for Online Handwriting Recognition », *Tenth International Conference on Document Analysis and Recognition (ICDAR 2009)*, p. 81-85, 2009.

Almaksour A., Mouchère H., Anquetil E., « Fast Online Incremental Learning with Few Examples For Online Handwritten Character Recognition », *Proceedings of the Eleventh International Conference on Frontiers in Handwriting Recognition (ICFHR'08)*, Montréal, Québec, Canada, p. 623-628, August, 2008.

Angelov P., Filev D., « An approach to online identification of Takagi-Sugeno fuzzy models », *Systems, Man, and Cybernetics, Part B : Cybernetics, IEEE Transactions on*, vol. 34, n° 1, p. 484-498, Feb., 2004.

Asuncion A., Newman D., « UCI Machine Learning Repository », 2007.

Carpenter G. A., Grossberg S., « The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network », *Computer*, vol. 21, n° 3, p. 77-88, 1988.

Carpenter G., Grossberg S., Markuzon N., Reynolds J., Rosen D., « Fuzzy ARTMAP : A neural network architecture for incremental supervised learning of analog multidimensional maps », *IEEE Transactions on Neural Networks*, 1992.

Chen K., Wang S., « Semi-supervised Learning via Regularized Boosting Working on Multiple Semi-supervised Assumptions », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

Chung F., Lee T., « Fuzzy Competitive Learning », *IEEE Transaction on Neural Network*, vol. 7, n° 3, p. 539-551, 1994.

De Backer S., Scheunders P., « Texture Segmentation by Frequency-Sensitive Elliptical Competitive Learning », *Image and Vision Computing*, vol. 19, n° 9–10, p. 639-648, 2001.

Gary G. Yen P. M., « An effective neuro-fuzzy paradigm for machinery condition health monitoring », *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 31-4, p. 523 - 536, 2001.

Jang J.-S., « ANFIS : adaptive-network-based fuzzy inference system », *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 23, p. 665-685, 1993.

Jr. J. J. L., Zeleznik R. C., « A Practical Approach for Writer-Dependent Symbol Recognition Using a Writer-Independent Symbol Recognizer », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, n° 11, p. 1917-1926, 2007.

Littlestone N., « Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow », *COLT '91 : Proceedings of the fourth annual workshop on Computational learning theory*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, p. 147-156, 1991.

Littlestone N., Warmuth M. K., « The weighted majority algorithm », *Inf. Comput.*, vol. 108, n° 2, p. 212-261, 1994.

Lughofer E., « FLEXFIS : A Robust Incremental Learning Approach for Evolving TakagiSugeno Fuzzy Models », *Fuzzy Systems, IEEE Transactions on*, vol. 16, n° 6, p. 1393-1410, Dec., 2008.

Lughofer E., Angelov P., Zhou X., « Evolving Single- And Multi-Model Fuzzy Classifiers with FLEXFIS-Class », *Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International*, p. 1 -6, 2007.

Minku F. L., Inoue H., Yao X., « Negative correlation in incremental learning », *Natural Computing : an international journal*, vol. 8, n° 2, p. 289-320, 2009.

Mouchere H., Anquetil E., « Synthèse de caractères manuscrits en-ligne pour la reconnaissance de l'écriture », *Actes du Colloque International Francophone sur l'Ecrit et le Document (CIFED'06)*, p. 187-192, 2006.

Mouchere H., Anquetil E., Ragot N., « On-line writer adaptation for handwriting recognition using fuzzy inference systems », *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, vol. 21(1), p. 99-116, 2007.

Moustakides G., « Study of the transient phase of the forgetting factor RLS », *Signal Processing, IEEE Transactions on*, vol. 45, n° 10, p. 2468 -2476, October, 1997.

Polikar R., Udpa L., Udpa S., Honavar V., « Learn++ : An Incremental Learning Algorithm for Supervised Neural Networks », *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 31, p. 497-508, 2001.

Reinke R., Michalski R., « Incremental learning of concept descriptions : A method and experimental results », *Hayes, J., Michie, D., Richards, J., eds. : Machine Intelligence*, vol. 11, p. 263288, 1988.

Sadri J., Suen C. Y., Bui T. D., « A New Clustering Method for Improving Plasticity and Stability in Handwritten Character Recognition Systems », *Pattern Recognition, International Conference on*, vol. 2, p. 1130-1133, 2006.

Siebert J. P., Vehicle Recognition Using Rule Based Methods, Technical report, Mar, 1987.

Takagi T., Sugeno M., « Fuzzy identification of systems and its applications to modeling and control », *IEEE TSMC*, vol. 15, n° 1, p. 116-132, 1985.

Yager R. R., Fileu D. P., « Learning of fuzzy rules by mountain clustering », vol. 2061, SPIE, p. 246-254, 1993.